

Fig. 1

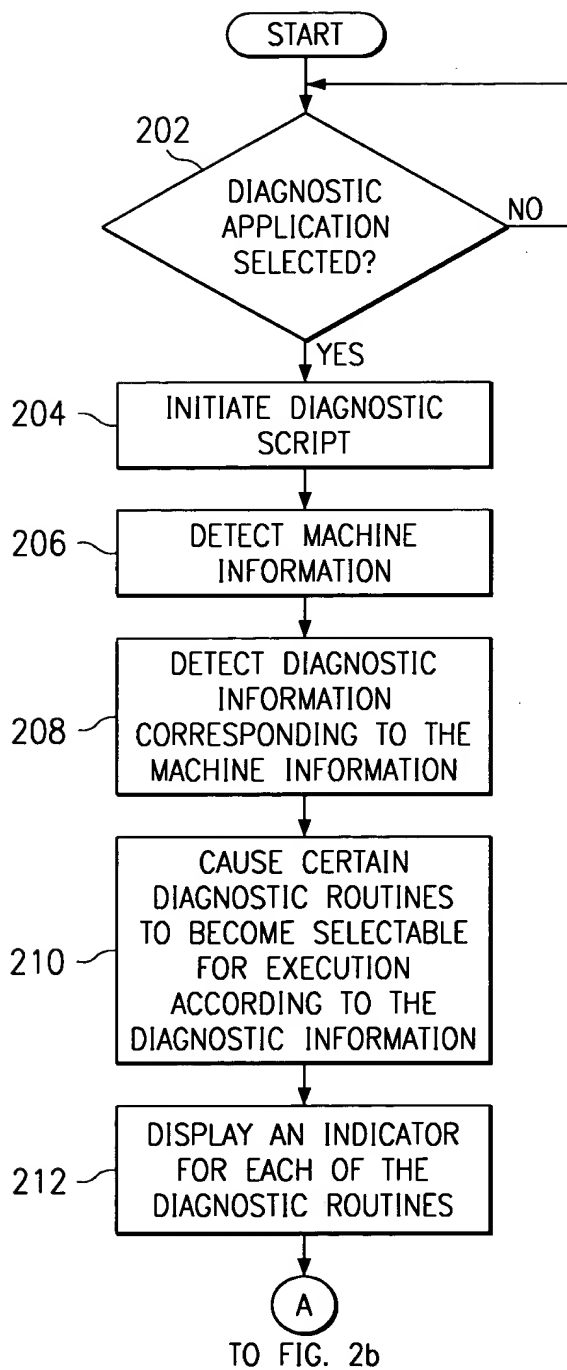


Fig. 2a

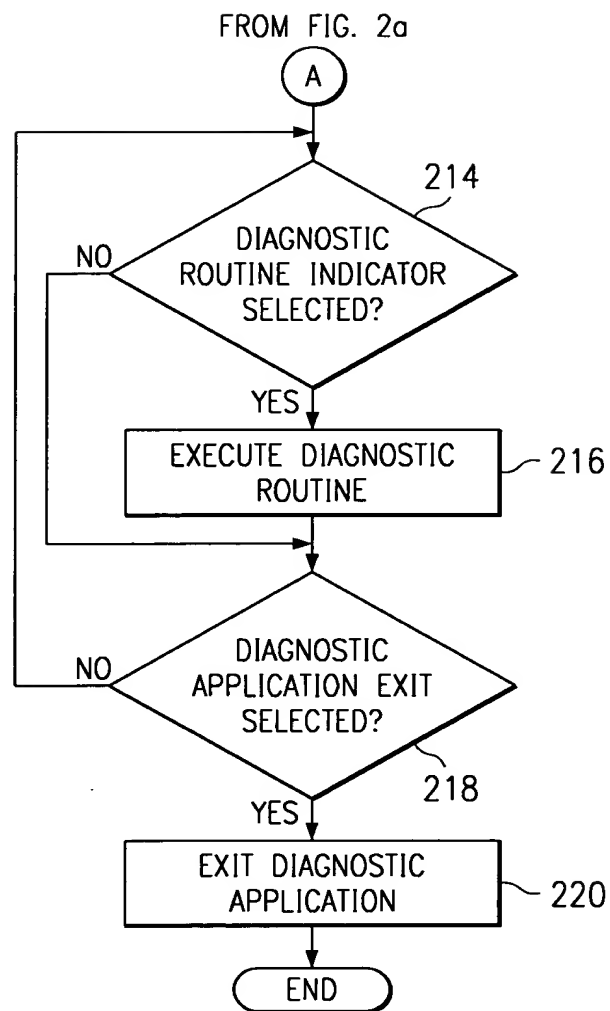


Fig. 2b

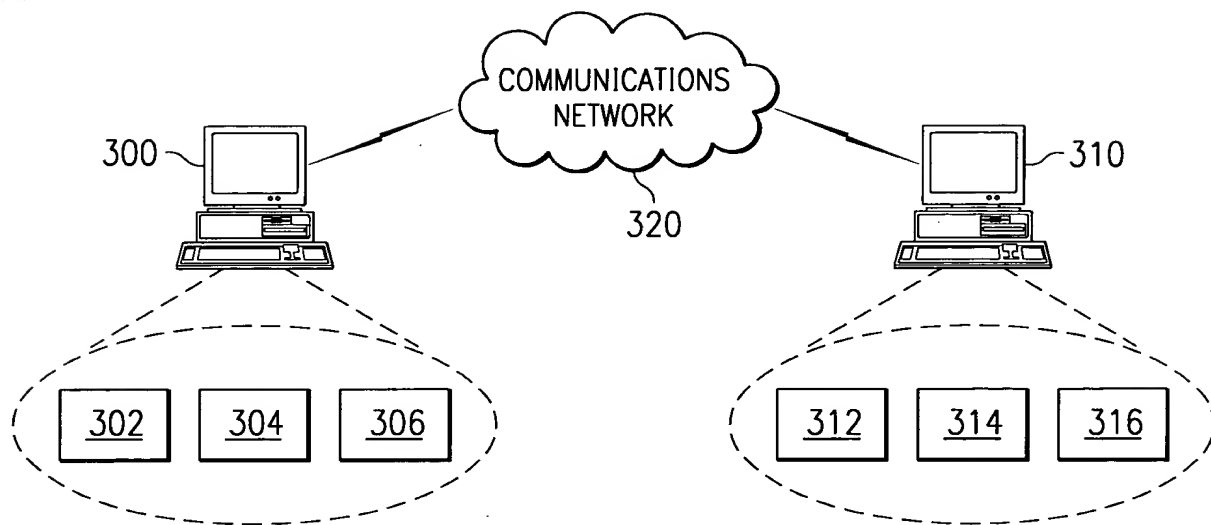


Fig. 3a

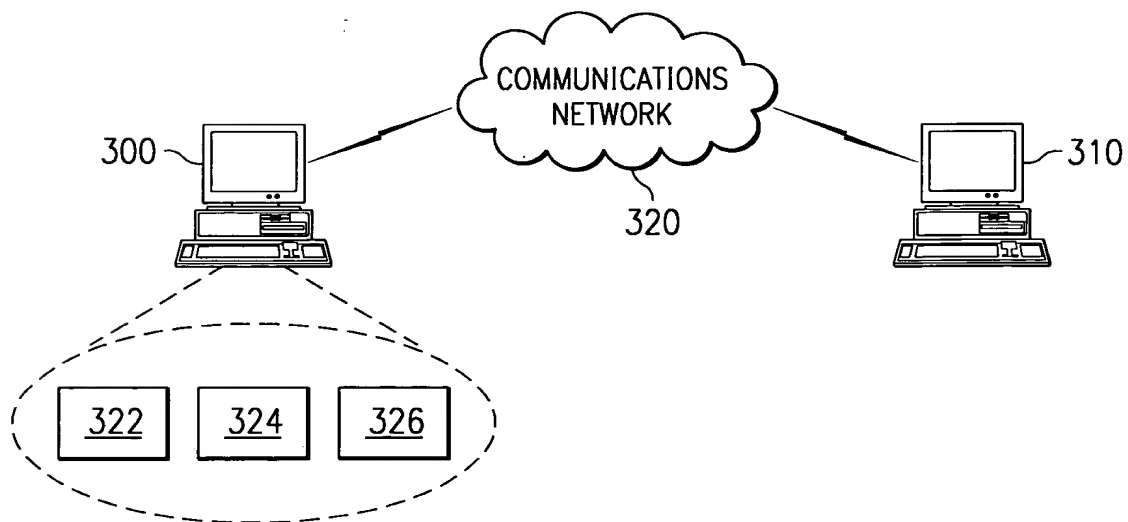


Fig. 3b

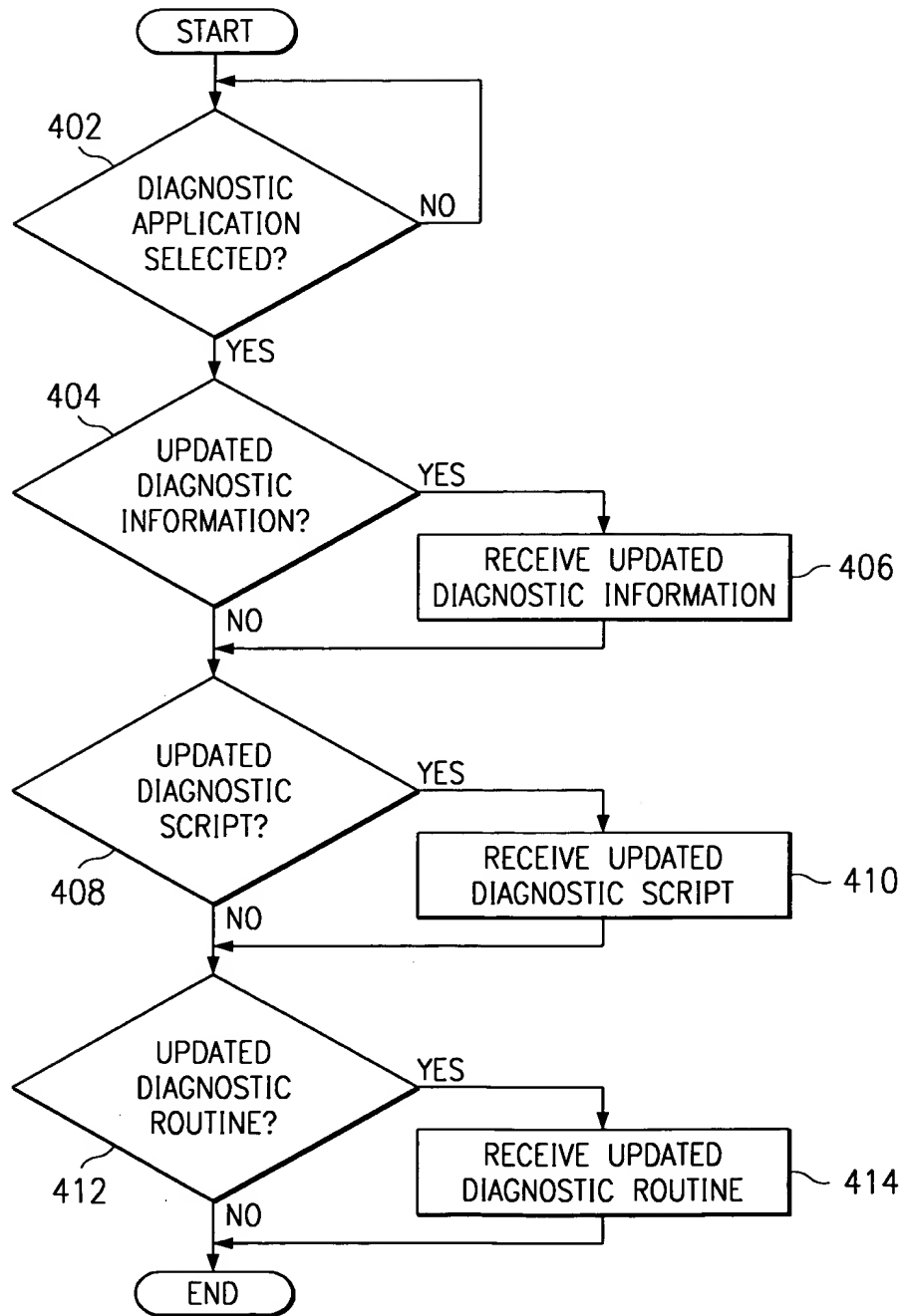


Fig. 4

DIAGCONF.INI File 500

[Win9x] ← 512

SupportedDiags=AGP|Audio|AVI|CD/DVD-ROM|CMOS|CPU|
Extended SMART Drive|Floppy Drive|Hard Drive|Infrared Port|Joystick|Keyboard|
Microphone|Modem|Monitor|Mouse|Network Adapter|Parallel Port|PCI|PCMCIA|
Printer|SCSI|Serial Port|SMART Drive|Super Disk|System Diagnosis|System Memory|
USB|Video Adapter|Zip Drive } 510

[WinNT] ← 522

SupportedDiags=AGP|Audio|AVI|CD/DVD-ROM|CMOS|CPU|Extended SMART Drive|
Floppy Drive|Hard Drive|Joystick|Keyboard|Microphone|Modem|Monitor|Mouse|
Network Adapter|Parallel Port|Printer|SCSI|Serial Port|SMART Drive|Super Disk|
System Diagnosis|System Memory|Tape Backup|Video Adapter|Zip Drive } 520

[Win2K] ← 532

SupportedDiags=AGP|Audio|AVI|CD/DVD-ROM|CMOS|CPU|Extended SMART Drive|
Floppy Drive|Hard Drive|Joystick|Keyboard|Microphone|Modem|Monitor|Mouse|
Network Adapter|Parallel Port|Printer|SCSI|Serial Port|SMART Drive|Super Disk|
System Diagnosis|System Memory|Tape Backup|Video Adapter|Zip Drive|USB } 530

[Hard Drive] ← 540

abstract=HardDisk Drive Diagnostic ← 542

icon=HardDrive.jpg ← 544

[0xBA] ← 550

;Code=Mummy

;Audience=Dimension

;Computer=Dimension Columbus II } 552

Win9xExclude=AVI|Extended SMART Drive|SMART Drive|Monitor|Infrared Port| ← 556

PCMCIA|Tape Backup

WinNTExclude=AVI|Extended SMART Drive|SMART Drive|Monitor|Infrared Port| ← 558

PCMCIA|PCI|USB

Win2KExclude=AVI|Extended SMART Drive|SMART Drive|Monitor|Infrared Port| ← 560

PCMCIA|PCI } 554

Fig. 5

Java Script/Methods/Functions: ← 600

```
function parseDiagIni (sMachineld)
{
    // *** Get Diagnostic Information File location. ← 602
    var slniFile = getDiagConfFile ();

    // *** Determine OS platform, a Java method is called which in turn calls a
    Windows API that returns the OS installed. ← 604
    var sOSSString = "";
    if (IsWindowsNT ())
        sOSSString = "WinNT";
    else if (IsWindows2000 ())
        sOSSString = "Win2K";
    else if (IsWindows98 () || IsWindows95 ())
        sOSSString = "Win9x";

    // *** Load list of supported diagnostics for the OS platform from the ← 606
    information file.
    var xSupportedDiags = iniApi.GetEntry (sOSSString, "SupportedDiags",
slniFile);
    xSupportedDiags.Print ();
    if (xSupportedDiags.GetReturnCode () != 0) {
        DebugPrint ("No SupportedDiags variable for " + sOSSString);
        return;
    }
    sSupportedDiags = xSupportedDiags.GetAttribute ("value");
    DebugPrint ("SupportedDiags = " + sSupportedDiags);
    var aSupportedDiags = parseLine (sSupportedDiags);
}
```

Fig. 6a

```

// *** Load list of unsupported diagnostics from the information file based
on machine and OS platform.
var xExcludeDiags = iniApi.GetEntry (sMachineld, sOSSString + "Exclude",
sIniFile);
    xExcludeDiags.Print ();
if (xExcludeDiags.GetReturnCode () == 0) {
    DebugPrint("Now entering - excluded diags section - svk");
    sExcludeDiags = xExcludeDiags.GetAttribute ("value");
    DebugPrint("ExcludeDiags = " + sExcludeDiags);
    var aExcludeDiags = parseLine (sExcludeDiags);
    // *** Remove the excluded diagnostics from the supported list.
    for (var i = 0; i < aExcludeDiags.length; i++) {
        for (var j = 0; j < aSupportedDiags.length; j++) {
            if (aExcludeDiags[i] == aSupportedDiags[j]) {
                aSupportedDiags[j] = "";
            }
        }
    }
}
else {
    DebugPrint ("No Exclude list for " + sOSSString + "on" + sMachineld);
}

```

Fig. 6b

```
// *** Build list of supported diagnostic and presentation items from the information file. ← 612
// Diagnostic|Abstract|Icon File Name||Diagnostic...
aDiagConfData = new Array ();
aDiagConfData[0] = "abstract";
aDiagConfData[1] = "icon";
var sFinalDiagsSupported = "";
var sSeparator = "|";
var sSuperSeparator = "||";
for (var j = 0; j < aSupportedDiags.length; j++) {
    DebugPrint ("Testing" + aSupportedDiags[j]);
    if ("" != aSupportedDiags[j]) {
        var Abstract = "";
        var Icon = "";
        var xDiagConfDataAbs = iniApi.GetEntry (aSupportedDiags[j],
aDiagConfData[0], sIniFile);
        if (xDiagConfDataAbs.GetReturnCode () == 0)
            Abstract = xDiagConfDataAbs.GetAttribute ("value");
        var xDiagConfDataIcon = iniApi.GetEntry (aSupportedDiags[j],
aDiagConfData[1], sIniFile);
        if (xDiagConfDataIcon.GetReturnCode () == 0)
            Icon = xDiagConfDataIcon.GetAttribute ("value");
        sFinalDiagsSupported = sFinalDiagsSupported + aSupportedDiags[j] +
sSeparator + Abstract + sSeparator + Icon + sSuperSeparator;
    }
}
// *** Set the environment global space ← 614
iapi.SetGlobalArg ("DellSupportedDiags", sFinalDiagsSupported);
}

// *** Load the machine id for the machine under test. ← 616
s = new java.lang.StringBuffer (200);

// *** Call a Java method that extracts the Machine ID from the SMBIOS table ← 618
DellAPI.GetMachineID (s);
if (s.toString ().length () > 0 ) {
    // Set the Machine Id in the following format: 0xFF. This is used to look the
    system up in DiagConf.ini
    g_sMachineId = "0x" + s.toString ();
}
// *** Create the list of supported diagnostics and their presentation items. ← 620
parseDiagIni (g_sMachineId);
```

Fig. 6c

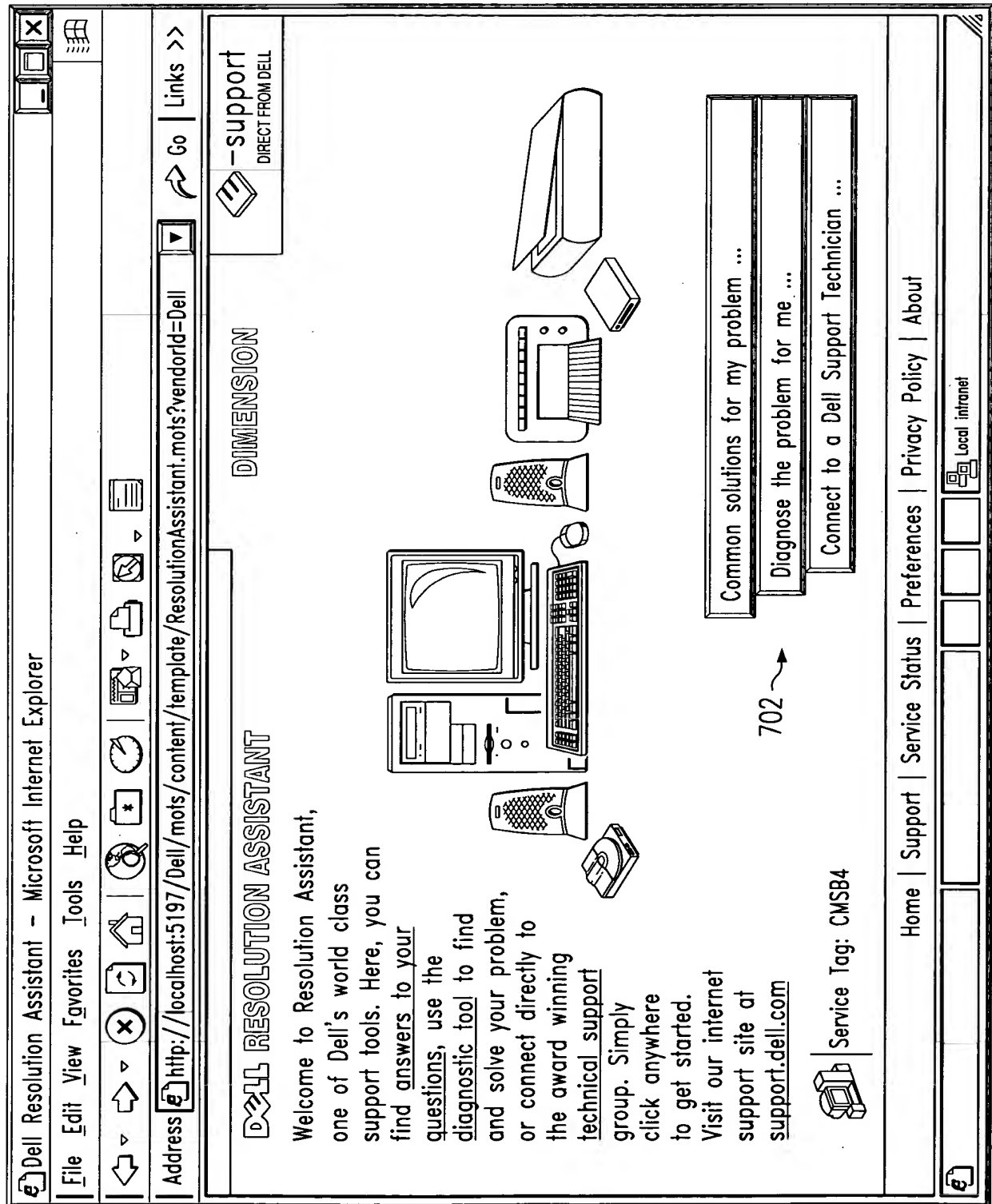


Fig. 7a

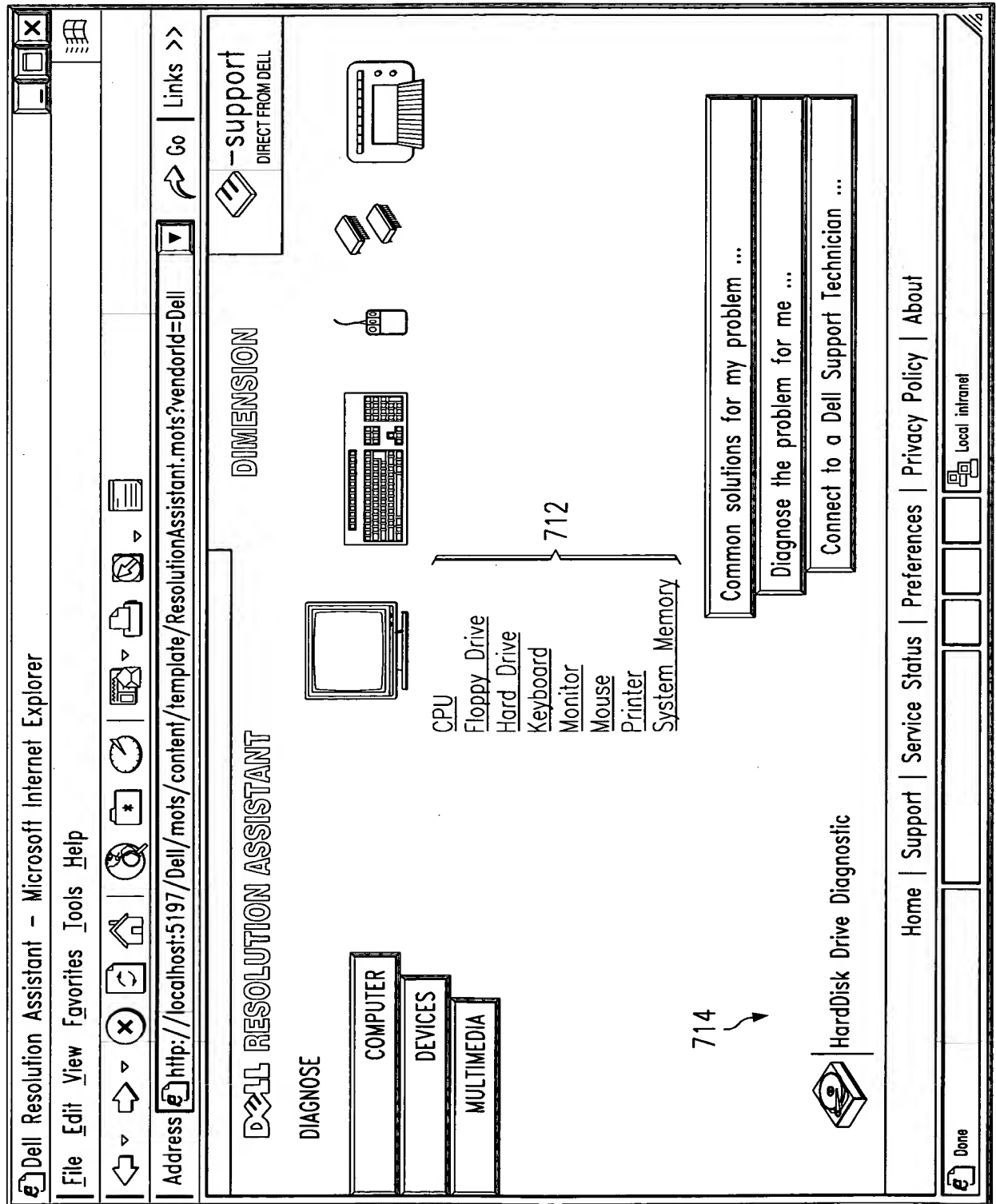


Fig. 7b

710

714

Fig. 7c

720

